Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Segundo Cuatrimestre, 30 de mayo de 2019. Grupos B y D

1. (2.5 puntos) "Estremecer" una lista consiste en colocar todos los elementos que aparecen en posiciones pares, por orden de aparición, seguidos de todos los que aparecen en posiciones impares, por orden inverso de aparición (primero el último, después el penúltimo, etc.). Por ejemplo, el resultado de "estremecer" a

0	1	2	3	4	5	6	7

es

0	2	4	6	7	5	3	1

(importante: consideramos que las posiciones comienzan en 0: es decir, la posición del primer elemento es la 0, la del segundo elemento es la 1, etc.)

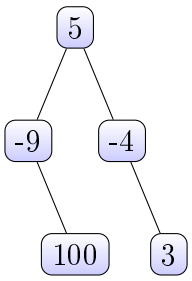
Añade una nueva operación mutadora

```
void estremece();
```

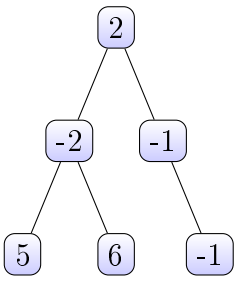
a la implementación del TAD `Lista` basada en nodos doblemente enlazados, que "estremezca" la lista, y determina justificadamente su complejidad. Dicha operación no puede invocar, ni directa ni indirectamente, operaciones de manejo de memoria dinámica (`new`, `delete`), ni tampoco puede realizar asignaciones a los contenidos de los nodos.

2. (2.5 puntos) Un "árbol de ganancias y pérdidas" es un árbol binario de enteros en el que los valores positivos de los nodos representan ganancias, mientras que los negativos representan pérdidas. Un nodo es "rentable" cuando todos sus ancestros son rentables, y, además, la suma de los valores de todos sus ancestros y el del suyo propio es positiva (dicha suma se denomina la "renta" del nodo). El "árbol de ganancias y pérdidas" es "rentable" cuando tiene alguna hoja "rentable" (la "renta" de dicha hoja se denomina una "renta" del árbol).

Ejemplos:



Arbol "rentable"; "renta" máxima: 4



Arbol no "rentable"

Debes programar el procedimiento:

```
void mejor_renta(Arbin<int> a,  bool & es_rentable, int & renta_maxima)
```

que determine: (i) si el "árbol de ganancias y pérdidas" `a` es "rentable" o no; y, en caso positivo, (ii) determine la "renta" máxima del árbol. Si el árbol es "rentable", tras finalizar la ejecución el parámetro `es_rentable` valdrá `true`, y `renta_maxima` contendrá la "renta" máxima del árbol (es decir, la mayor "renta" de las hojas que son rentables). Si el árbol no es "rentable", tras finalizar la ejecución el parámetro `es_rentable` valdrá `false`. En este caso, el valor de `renta_maxima` es irrelevante. También debes determinar justificadamente la complejidad del procedimiento.

3. (5 puntos) La tienda `Outlet aGoGo` nos ha encargado programar un TAD para gestionar su sistema de ofertas online. Este sistema permite ofertar cantidades limitadas de sus productos a precios especiales. Cada vez que un cliente desea beneficiarse de una oferta, el sistema lo pone en la lista de espera para dicha oferta, gestionando las ventas por estricto orden de llegada, hasta que se agoten las unidades disponibles (un cliente podrá esperar simultáneamente para comprar más de un producto). El TAD deberá incluir las siguientes operaciones:

- `crea`: Crea un sistema de venta vacío.
- `an_oferta(producto, num_unidades)`: Crea una nueva oferta para el producto `producto`, con `num_unidades` unidades disponibles. Esta es una operación parcial: no debe existir ya un producto con el mismo nombre en el sistema, y, además, el número de unidades debe ser positivo.
- `pon_en_espera(cliente, producto)`: Añade al cliente `cliente` a la lista de espera para la compra del producto en oferta `producto`. En caso de que el cliente ya esté esperando para comprar dicho producto, la operación no tendrá ningún efecto. Esta es una operación parcial: el producto al que se hace referencia debe estar ofertándose actualmente en el sistema.
- `cancela_espera(cliente, producto)`: Elimina al cliente `cliente` de la lista de espera del producto en oferta `producto`. Si el cliente no está esperando en la lista de espera del producto, la operación no tendrá ningún efecto. Esta es una operación parcial: el producto debe estar ofertándose en el sistema.
- `num_en_espera(producto)`→`num_clientes`: Devuelve el número de clientes que están esperando para comprar el producto `producto`. Esta es una operación parcial: el producto debe estar ofertándose en el sistema.
- `venta(producto, num_unidades)`: Registra una venta de `num_unidades` unidades del producto `producto` al primer cliente de la lista de espera. Dicho cliente se elimina de la lista de espera. En caso de que, tras dicha venta, no queden más unidades, la venta para el producto se cerrará, eliminado la oferta del sistema (y, por tanto, todos los clientes que están esperando se quedarán sin producto). Esta es una operación parcial: el producto debe estar ofertándose actualmente en el sistema, la lista de espera para dicho producto no debe estar vacía, y el número de unidades solicitado no debe sobrepasar el número de unidades disponibles.
- `primero_en_espera(producto)`→`cliente`: Devuelve el nombre del primer cliente que está esperando para comprar el producto `producto`. Esta es una operación parcial: el producto se debe estar ofertando actualmente, y su lista de espera no debe estar vacía.
- `lista_ventas()`→`Lista`. Devuelve una lista del número de unidades vendido para cada producto desde la puesta en marcha del sistema. Cada elemento de esta lista consiste en: (i) el nombre del producto; y (ii) todas las unidades vendidas de este producto (si un producto se ha ofertado varias veces, este valor será el total vendido para cada oferta). La lista estará ordenada alfabéticamente por los nombres de productos.

Aparte de realizar la implementación del TAD, debes indicar justificadamente cuál es la complejidad de cada operación. Al tratarse de un sistema crítico, la implementación deben ser lo más eficientes posible.